Package: clustur (via r-universe)

November 23, 2026

Type Package

Title Clustering

Version 0.2.1

Date 2026-11-17

Maintainer Patrick Schloss <pschloss@umich.edu>

Description A tool that implements the clustering algorithms from 'mothur' (Schloss PD et al. (2009) <doi:10.1128/AEM.01541-09>). 'clustur' make use of the cluster() and make.shared() command from 'mothur'. Our cluster() function has five different algorithms implemented: 'OptiClust', 'furthest', 'nearest', 'average', and 'weighted'. 'OptiClust' is an optimized clustering method for Operational Taxonomic Units, and you can learn more here, (Westcott SL, Schloss PD (2017) <doi:10.1128/mspheredirect.00073-17>). The make.shared() command is always applied at the end of the clustering command. This functionality allows us to generate and create clustering and abundance data efficiently.

License GPL (>= 3)

Encoding UTF-8

Imports Matrix, methods, Rcpp (>= 1.0.12), utils, reshape2, strollur

Depends R (>= 4.2.0)

LinkingTo Rcpp, RcppThread, sitmo, testthat

Suggests knitr, rmarkdown, testthat (>= 3.0.0), xml2

Config/testthat/edition 3

VignetteBuilder knitr

URL http://www.schlosslab.org/clustur/, https://github.com/SchlossLab/clustur, https://github.com/SchlossLab/Clustur

BugReports https://github.com/SchlossLab/Clustur/issues

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Gregory Johnson [aut] (ORCID:
<https://orcid.org/0009-0008-3890-0297>), Sarah Westcott [aut],
Patrick Schloss [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-6935-4275>)

Repository https://cran.r-universe.dev

Date/Publication 2026-11-23 08:10:43 UTC

RemoteUrl https://github.com/cran/clustur

RemoteRef HEAD

RemoteSha 193785a2f32df97bf4ab49d8276f93027fa55723

Contents

cluster . . . 2
cluster_split . . . 4
create_sparse_matrix . . . 6
example_path . . . 7
get_abundance . . . 8
get_bins . . . 8
get_count_table . . . 9
get_cutoff . . . 10
get_distance_df . . . 10
get_metrics . . . 11
optifit . . . 12
read_count . . . 14
read_dist . . . 15
split_clusters_to_list . . . 16
validate_count_table . . . 17

Index **18**

`cluster` *Cluster entities together*

Description

Clusters entities represented in a distance matrix and count table using one of several algorithms and outputs information about the composition and abundance of each cluster

Usage

```
cluster(
  distance_object,
  cutoff,
  method = "opticlust",
  feature_column_name_to = "feature",
  bin_column_name_to = "bin",
  random_seed = 123,
  delta = 1e-04,
  max_iterations = 100,
  initialize = "singleton",
  metric = "mcc",
  strollur_object = NULL
)
```

Arguments

distance_object	The distance object that was created using the 'read_dist()' function.
cutoff	The cutoff you want to cluster towards.
method	The method of clustering to be performed: opticlust (default), furthest, nearest, average, or weighted.
feature_column_name_to	Set the name of the column in the cluster dataframe that contains the sequence names.
bin_column_name_to	Set the name of the column in the cluster dataframe that contains the name of the group of sequence names.
random_seed	the random seed to use, (default = 123).
delta	Allows you to set the stable value for the metric in the opticluster method. It defaults to 0.0001, to reach full convergence set this value to 0.
max_iterations	the max amount of iterations the opticluster method will run. The default is 100.
initialize	this parameter allows you to set the default randomization of for the opticluster method. The two methods are: "singleton" and "oneotu." Singleton will randomly assign each sequence to its own OTU while "oneotu" assigns all sequence into one OTU. Defaults to singleton.
metric	this parameter allows you to choose which metric the cluster will optimize to, the options are: Matthews correlation coefficient (mcc), sensitivity (sens), specificity (spec), true positives + true negatives (tptn), false positives + false negatives (fpfn), true positives (tp), true negative (tn), false positive (fp), false negative (fn), f1score (f1score), accuracy (accuracy), positive predictive value (ppv), negative predictive value (npv), false discovery rate (fdr). The default metric is mcc.
strollur_object	Using the 'strollur' r package, you can use your created strollur object to save your clustering results to. This allows you to seamless modify your strollur object and analysis results as needed.

Value

A list of ‘data.frames‘ that contain abundance, and clustering results. If you used ‘method = opticlust‘, it will also return clustering performance metrics.

Examples

```
cutoff <- 0.03
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff)

cluster_results <- cluster(distance_data,
                           cutoff, method = "opticlust",
                           feature_column_name_to = "sequence",
                           bin_column_name_to = "omu")
cluster_results <- cluster(distance_data,
                           cutoff, method = "furthest")
cluster_results <- cluster(distance_data,
                           cutoff, method = "nearest")
cluster_results <- cluster(distance_data,
                           cutoff, method = "average")
cluster_results <- cluster(distance_data,
                           cutoff, method = "weighted")
```

`cluster_split` *Cluster Split*

Description

Cluster Split splits data into groups based on taxomic group. It then clusters them using the following methods: opticluster, average neighbor, nearest neighbor, furthest neighbor, and weighted neighbor. You can read more about [cluster split](https://mothur.org/wiki/cluster.split/) on the mothur wiki.

Usage

```
cluster_split(
  fasta,
  taxonomy,
  count_table,
  cluster_method,
  cutoff,
  taxonomy_cutoff,
  feature_column_name = "feature",
  bin_column_name = "bin",
  random_seed = 123,
```

```
  delta = 1e-04,
  max_iterations = 100,
  initialize = "singleton",
  metric = "mcc",
  pairwise_distance_method = "onegap",
  number_of_threads = 1,
  strollur_object = NULL
)
```

Arguments

fasta	a‘data.frame‘ representation of your fasta data. This can be generated using the ‘read_fasta‘ function. The fasta ‘data.frame‘ must contain two columns: sequence_name and sequence.
taxonomy	a‘data.frame‘ representation of your taxonomy data. This can be generated using the ‘read_mothur_taxonomy‘ function. The taxonomy ‘data.frame‘ must contain two columns: sequence_name and taxonomy.
count_table	the count table that represents the data, must be read using the ‘read_count()‘ function.
cluster_method	the clustering method you wish to use. Your clustering options are: opticluster, average neighbor, nearest neighbor, furthest neighbor, and weighted neighbor.
cutoff	a ‘numerical‘ value that represents the distance cutoff.
taxonomy_cutoff	a ‘integer‘ value that represents the taxonomic level cutoff.
feature_column_name	Set the name of the column in the cluster dataframe that contains the sequence names. This is a ‘character‘ value.
bin_column_name	Set the name of the column in the cluster dataframe that contains the name of the group of sequence names. This is a ‘character‘ value.
random_seed	a ‘integer‘ value that represents the current random seed.
delta	Allows you to set the stable value for the metric in the opticluster method. It defaults to 0.0001, to reach full convergence set this value to 0.
max_iterations	the max amount of iterations the opticluster method will run. The default is 100.
initialize	this parameter allows you to set the default randomization of for the opticluster method. The two methods are: "singleton" and "oneotu." Singleton will randomly assign each sequence to its own OTU while "oneotu" assigns all sequence into one OTU. Defaults to singleton.
metric	this parameter allows you to choose which metric the cluster will optimize to, the options are: Matthews correlation coefficient (mcc), sensitivity (sens), specificity (spec), true positives + true negatives (tptn), false positives + false negatives (fpfn), true positives (tp), true negative (tn), false positive (fp), false negative (fn), f1score (f1score), accuracy (accuracy), positive predictive value (ppv), negative predictive value (npv), false discovery rate (fdr). The default metric is mcc.

`pairwise_distance_method`
: the method you wish to use to calculate pairwise distances between aligned sequences. As of current we only allow the calculation of fasta sequences, and the current methods are; onegap, nogap, and eachgap. For more information look at the (mother webpage)[https://mothur.org/wiki/dist.seqs/]. This is a 'character' value.

`number_of_threads`
: a 'integer' value that represents the amount of allotted threads that can be used.

`strollur_object`
: Using the 'strollur' r package, you can use your created strollur object to save your clustering results to. This allows you to seamless modify your strollur object and analysis results as needed.

Value

A 'list' of 'data.frames' that contain abundance and clustering results, and clustering performance metrics.

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
fasta <- read_fasta(example_path("amazon.align"))
taxonomy <- read_mothur_taxonomy(example_path("amazon.taxonomy"))
df <- cluster_split(fasta, taxonomy, count_table, "opti",
                    cutoff, 3, number_of_threads = 1)
```

---

create_sparse_matrix *Create Sparse Matrix*

Description

Given a list of i indexes, j indexes, and distances values, we can create a sparse distance matrix for you. Each vector must have the same size.

Usage

```
create_sparse_matrix(i_index, j_index, distances)
```

Arguments

`i_index`	A list of i indexes, must be numeric
`j_index`	A list of j indexes, must be numeric
`distances`	A list of the distance at the i and j index

Value

a ‘dgTMatrix‘ from the ‘Matrix‘ library.

Examples

```
i_values <- as.integer(1:100)
j_values <- as.integer(sample(1:100, 100, TRUE))
x_values <- as.numeric(runif(100, 0, 1))
s_matrix <- create_sparse_matrix(i_values, j_values, x_values)
```

`example_path` *Example Path*

Description

This function was created as a helper function to generate file paths to our internal data. You should use this function if you want to follow along with the example, or interact with the data

Usage

```
example_path(file = NULL)
```

Arguments

`file`	The file name of the data; leave as NULL (default) to get full list of example files

Value

the path to the file as a ‘character‘ or a vector of ‘character‘ giving example filenames if ‘fill = NULL‘.

Examples

```
example_path("amazon_phylip.dist")
example_path()
```

get_abundance *Get Shared*

Description

GetShared returns the generated abundance ‘data.frame‘ from the ‘cluster()‘ function

Usage

```
get_abundance(cluster_data)
```

Arguments

cluster_data	The output from the ‘cluster()‘ function.

Value

a shared data.frame

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
df_clusters <- cluster(distance_data, cutoff, method = "opticlust")
shared <- get_abundance(df_clusters)
```

get_bins *Get Clusters*

Description

GetClusters returns a ‘data.frame‘ of the generated clusters from the ‘cluster()‘ function.

Usage

```
get_bins(cluster_data)
```

Arguments

cluster_data	The output from the ‘cluster()‘ function.

Value

the created cluster ‘data.frame‘.

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
df_clusters <- cluster(distance_data, cutoff, method = "opticlust")
clusters <- get_bins(df_clusters)
```

`get_count_table` *Get Count Table*

Description

This function returns the count table that was used to generate the distance object.

Usage

```
get_count_table(distance_object)
```

Arguments

`distance_object`
: The output from the ‘read.dist()‘ function.

Value

a count_table ‘data.frame‘.

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
count_table <- get_count_table(distance_data)
```

get_cutoff *Get Cutoff*

Description

Returns the distance cutoff of the cluster object from the ‘cluster()‘ function

Usage

```
get_cutoff(cluster_data)
```

Arguments

cluster_data	The output from the ‘cluster()‘ function.

Value

the cutoff value as a ‘dbl‘

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
df_clusters <- cluster(distance_data, cutoff, method = "opticlust")
cutoff <- get_cutoff(df_clusters)
```

get_distance_df *Get Distance Data Frame*

Description

This function will generate a ‘data.frame‘ that contains the distances of all the indexes.

Usage

```
get_distance_df(distance_object)
```

Arguments

distance_object	The output from the ‘read.dist()‘ function.

Value

a distance ‘data.frame‘.

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                                        count_table, cutoff, FALSE)
count_table <- get_count_table(distance_data)
```

`get_metrics` *Get Metrics*

Description

GetMetrics returns the generated metrics ‘data.frame‘ from the ‘cluster()‘ function.

Usage

```
get_metrics(cluster_data)
```

Arguments

`cluster_data`	The output from the ‘cluster()‘ function.

Value

a list of metric data.frames

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
df_clusters <- cluster(distance_data, cutoff, method = "opticlust")
list_of_metrics <- get_metrics(df_clusters)
```

`optifit` *Optifit*

Description

Optifit is the 'cluster.fit' command in mothur. It allows you to fit sequences to OTUs or existing OTUS. Just like mothur, there are two ways of running this command: open, or closed. Closed fits reads into existing OTUs, and scraps any reads that cannot be fitted(default it closed). Open fits reads into existing OTUs while unfitted OTUs are clustered into new OTUs.

Usage

```
optifit(
  fit_dist,
  cutoff,
  feature_column_name = "feature",
  bin_column_name = "bin",
  accnos = NULL,
  ref_fasta = NULL,
  ref_count = NULL,
  ref_list = NULL,
  fit_fasta = NULL,
  fit_percent = 50,
  closed = FALSE,
  print_ref = TRUE,
  random_seed = 123,
  max_iterations = 100,
  denovo_iters = 10,
  optimal_denovo_result = TRUE,
  ref_weight = "none",
  metric = "mcc",
  pairwise_distance_method = "onegap",
  number_of_threads = 1,
  strollur_object = NULL
)
```

Arguments

`fit_dist`	The distance object that will be used to fit to the new reference. This must be created using the 'read_dist()' function.
`cutoff`	The cutoff you want to cluster towards.
`feature_column_name`	Set the name of the column in the cluster dataframe that contains the sequence names. This is a 'character' value.
`bin_column_name`	Set the name of the column in the cluster dataframe that contains the name of the group of sequence names. This is a 'character' value.

accnos — a ‘vector‘ of strings that represent the reference sequences by name. Use this when you know the group of sequences you want your fit data to fit against.

ref_fasta — a‘data.frame‘ representation of your fasta data for your reference sequences. This can be generated using the ‘read_fasta()‘ function. The fasta ‘data.frame‘ must contain two columns: "sequence_name" and "sequence."

ref_count — a ‘data.frame‘ that display sequence abundance for the reference sequences. Must be generated using the ‘read_count()‘ function.

ref_list — a ‘data.frame‘ respentation of the mothur list file you wish to fit your sequences to. Can be generated using the ‘read_mothur_list()‘ function. Must include two columns: "bin_name" and "sequence_name."

fit_fasta — a‘data.frame‘ representation of your fasta data for your fitted sequences. This can be generated using the ‘read_fasta()‘ function. The fasta ‘data.frame‘ must contain two columns: "sequence_name" and "sequence."

fit_percent — allows you to set the percentage of reads that need to be fitted. This is a ‘numerical‘ value.

closed — deteremines whether you want the function to run using the closed or open method. The closed method will return reads that fit into existing otus and scrap those that did not. The open method will keep reads that fit into existing otus and create new OTUs for those that did not. This is a ‘logical‘ value.

print_ref — controls whether all the reference sequences are printed in the final result. This is a ‘character‘ value.

random_seed — a ‘integer‘ value that represents the current random seed.

max_iterations — the max number of iterations that cluster computes. This is a ‘integer‘ value.

denovo_iters — the max number of iterations the denovo method computes. This is a ‘integer‘ value.

optimal_denovo_result — determines whether or not you want the denovo method to return the cluster with the highest mcc value or all of the randomly computed clusters. This is a ‘logical‘ value.

ref_weight — Ref Weight is used to randomize the reference sequences for the denovo method. There are three methods: none, abundance and connectivity. "none" will do a normal shuffle of the given sequences without any weights, abundance will shuffle the sequences using abundances as a weight, and connectivity will shuffle the sequences but uses a weight of one for every sequence.

metric — this parameter allows you to choose which metric the cluster will optimize to, the options are: Matthews correlation coefficient (mcc), sensitivity (sens), specificity (spec), true positives + true negatives (tptn), false positives + false negatives (fpfn), true positives (tp), true negative (tn), false positive (fp), false negative (fn), f1score (f1score), accuracy (accuracy), positive predictive value (ppv), negative predictive value (npv), false discovery rate (fdr). The default metric is mcc.

pairwise_distance_method — the method you wish to use to calculate pairwise distances between aligned sequences. As of current we only allow the calculation of fasta sequences, and the current methods are; onegap, nogap, and eachgap. For more information look at

the (mother webpage)[https://mothur.org/wiki/dist.seqs/]. This is a ‘character‘ value.

number_of_threads
: a ‘integer‘ value that represents the amount of allotted threads that can be used.

strollur_object
: Using the ‘strollur‘ r package, you can use your created strollur object to save your clustering results to. This allows you to seamless modify your strollur object and analysis results as needed.

Value

If used with the denovo method, it will return a ‘list‘ of ‘list‘ of ‘data.frames‘ that contain abundance and clustering results, and clustering performance metrics. If used with any other method (or with the optimal_denovo_result = T parameter), it will return a ‘list‘ of ‘data.frames‘ that contain abundance and clustering results, and clustering performance metrics.

Examples

```
amazon_count <- read_count(example_path("amazon.full.count_table"))
amazon_dist <- read_dist(example_path("amazon_phylip.dist"),
                         amazon_count, 0.03)

esophagus_fasta <- read_fasta(example_path("esophagus.fasta"))
esophagus_count <- read_count(example_path("esophagus.count_table"))
amazon_fasta <- read_fasta(example_path("amazon.align"))
esophagus_list <- read_mothur_list(example_path("esophagus.list"))


denovo <- optifit(amazon_dist, 0.03)
denovo_open <- optifit(amazon_dist, 0.03, close = FALSE)
accnos <- optifit(amazon_dist, 0.03,
                  accnos = amazon_count$Representative_Sequence[1:40])
accnos_open <- optifit(amazon_dist, 0.03,
                       accnos = amazon_count$Representative_Sequence[1:40],
                       closed=FALSE)

ref_cluster <- optifit(amazon_dist, 0.03, "feature", "bin", accnos = NULL,
                       ref_fasta = esophagus_fasta,
                       ref_count = esophagus_count,
                       ref_list = esophagus_list, fit_fasta = amazon_fasta,
                       pairwise_distance_method = "onegap",
                       number_of_threads = 1,
                       print_ref = TRUE)
```

---

read_count *Read count table*

Description

This function will read and return your count table. It can take in sparse and full count tables.

Usage

```
read_count(count_table)

## S3 method for class 'strollur'
read_count(count_table)

## S3 method for class 'character'
read_count(count_table)
```

Arguments

count_table	The file path of your count table or a strollur object.

Value

a count table ‘data.frame‘.

Examples

```
count_table <- read_count(example_path("amazon.full.count_table"))
```

---

read_dist *Read distance matrices*

Description

Read in distances from a file that is formatted with three columns for the row, column, and distance of a sparse, square matrix or in a phylip-formatted distance matrix.

Usage

```
read_dist(distance_file, count_table, cutoff, is_similarity_matrix = FALSE)
```

Arguments

distance_file	Either a phylip or column distance file, or a sparse matrix. The function will detect the format for you.
count_table	A table of names and the given abundance per group. Can be in mothur's sparse or full format. The function will detect the format for you.
cutoff	The value you wish to use as a cutoff when clustering.
is_similarity_matrix	are you using a similarity matrix (default) or distance matrix?

Value

A distance ‘externalptr‘ object that contains all your distance information. Can be accessed using ‘get_distance_df()‘

Examples

```
i_values <- as.integer(1:100)
j_values <- as.integer(sample(1:100, 100, TRUE))
x_values <- as.numeric(runif(100, 0, 1))
s_matrix <- create_sparse_matrix(i_values, j_values, x_values)

sparse_count <- data.frame(
                Representative_Sequence = 1:100,
                total = rep(1, times = 100))

column_path <- example_path("amazon_column.dist")
phylip_path <- example_path("amazon_phylip.dist")
count_table <- read_count(example_path("amazon.full.count_table"))

data_column <- read_dist(column_path, count_table, 0.03)
data_phylip <- read_dist(phylip_path, count_table, 0.03)
data_sparse <- read_dist(s_matrix, sparse_count, 0.03)
```

---

split_clusters_to_list

Split Clusters to List

Description

'split_clusters_to_list()' will extract clusters from the cluster generated 'data.frame'. It will then turn those clusters into a list. This allows users to more easily visualize their data.

Usage

```
split_clusters_to_list(cluster)
```

Arguments

cluster	The output generated from the 'cluster()' function.

Value

a named 'list' of clusters.

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
cluster_results <- cluster(distance_data, cutoff, method = "opticlust")

cluster_list <- split_clusters_to_list(cluster_results)
```

validate_count_table *Validate Count Table*

Description

If the count table is already valid nothing will change, otherwise it will add a new group to the count table file.

Usage

```
validate_count_table(count_table_df)
```

Arguments

count_table_df The count table 'data.frame' object.

Details

Determines whether user supplied count table is valid

Value

A validated count table 'data.frame'

Examples

```
count_table <- read.delim(example_path("amazon.full.count_table"))
count_table_valid <- validate_count_table(count_table)
```

Index

cluster, 2
cluster_split, 4
create_sparse_matrix, 6

example_path, 7

get_abundance, 8
get_bins, 8
get_count_table, 9
get_cutoff, 10
get_distance_df, 10
get_metrics, 11

optifit, 12

read_count, 14
read_dist, 15

split_clusters_to_list, 16

validate_count_table, 17